REMARKS

This is a reply to Examiner's Office communication of May 5, 2006.

Rejections Under 35 U.S.C. 101

Claims 1-8, 10-14, 17 and 18 are rejected 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter.

In order to address these rejections, in claim 1 "A method for detecting cross-iteration dependencies between variables in a loop of a computer program, the method comprising the steps of" has been amended to read - - A method for executing, by a processor of a computer system, a set of program instructions for a loop, wherein the executing comprises performing the steps of - - . Further, "associating unique values with each of the values of indirect loop index variables of the loop" has been amended to read - - storing in a tangible, computer-readable storage media a set of unique proxy values, the unique proxy values being substituted for respective values of indirect loop index variables of the loop - -. Further, the claim is amended to state that the method includes "executing the set of program instructions for the loop by the processor, wherein ones of the iterations are executed concurrently responsive to determining that no cross-iteration dependencies exist between the ones of the iterations of the loop based upon the indirectly indexed access patterns of the ones of the iterations." Thus claim 1, as amended, includes language regarding executing instructions for a particular useful purpose in a computer system, in a fashion similar to original claim 9, which was found by the Examiner to be acceptable under 35 U.S.C. 101.

Claims 2-5 and 8-16 are canceled. Claims 17 and 18 are amended to include language similar to claim 1.

Rejections Under 35 U.S.C. 112 second paragraph

Claims 13 and 15 are rejected under 35 U.S.C. 112 second paragraph as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention. Applicant herein cancels claims 13 and 15.

Rejections Under 35 U.S.C. 102 (b)

Claims 1, 2, 8-11, and 14-18 are rejected 35 U.S.C. 102(b) as being anticipated by Huang et al., "Non-linear array data dependence test" ("Huang"). Claims 1, 17 and 18 are herein amended to overcome the rejections. Claims 2-5 and 8-16 are canceled. Claims 19-21 are added to further distinguish the invention.

Claims 1, 17 and 18

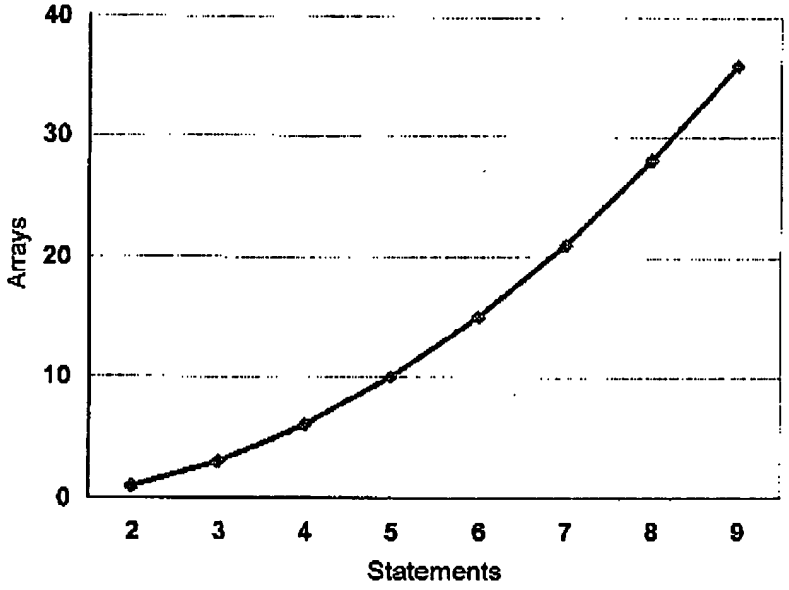
In the above amendments described above regarding 35 U.S.C. 101, the adjective "proxy" has been added to modify the "set of unique values" associated with respective values of indirect loop index variables of the loop. This is partly to distinguish between later references in the claim to these values, i.e., so that a later reference may more easily point out which values are the subject of the reference. It is also to particularly point out that the set of unique values serve as proxies for the values of the indirect loop index variables. While the term "proxy" may is not used in haec verba in the specification, no new matter is added and the written description requirement is met,[1] since it is clear from the context of the specification that a certain set of unique values are substituted for, and thereby serve as proxies for, the values of the indirect loop index variables. See present application, page 10, lines 28-29 ("The use of prime numbers *in place of* the indirect loop index values allows a group of such index values to be represented by a unique number." emphasis added).

For each respective potential dependency among statements, Huang generates an array. Huang, page 147, section 3, third paragraph ("For each potential dependency $S_i \delta S_j$, where S_i occurs textually before S_j and at least one is *define*, we create another array T . . ."') It should be understood that this may give rise to a relatively large number of arrays for a relatively small number of statements. For example, it follows from the above statement by Huang that for four statements in a loop body, Huang's method may give rise to as many as seven arrays. More generally, for N statements in a loop body, Huang's method may give rise to as many as the following number of T arrays: $1 + 2 + 3 + \ldots + (N-1)$.

---

[1] The issue with the written description requirement is not whether the claimed subject matter is described "in haec verba" in the specification, but merely whether it is described in a way "so that one skilled in the art can recognize what is claimed." University of Rochester v. G.D. Searle & Co., Inc., United States Court of Appeals for the Federal Circuit, 03-1304, decided February 13, 2004 (citing Enzo Biochem, Inc. v. Gen-Probe Inc., 323 F.3d 956, 968 (Fed. Cir. 2002)).

Expressed graphically, Huang's method may give rise to numbers of *T* arrays as shown below:



It also important to understand that while the *T* arrays taught by Huang might be characterized as a type of access pattern, they are not the "indirectly indexed access pattern" disclosed in the present application. And it is important to understand that efficiency advantages are gained by the method, system and computer program product claimed in the present application, which includes use of an indirectly indexed access pattern that is different than the *T* arrays taught by Huang. In order to more clearly distinguish the present invention from the teachings of Huang in this regard, Applicant herein amends claims 1, 10, 11, 17 and 18 as described below.

According to one specifically described embodiment of the present application, an indirectly indexed access pattern for a loop may be a set of a maximum of three arrays, regardless of the number of statements in the loop body. See present application, page 10, lines 23 -29 (describing arrays S_A, S_T and S_P). The elements of S_A include values generated responsive to indirect loop indices for *all* "active array variables," where the loop may include a boolean condition and the set of all active array variables is the set of array variables defined in

assignment statements of the loop body and active either if the boolean condition evaluates to be true or if the boolean condition evaluates to be false. See present application, page 9, lines 7 - 31, and equation for array S_A at page 10, line 24. The elements of S_T include values generated responsive to indirect loop indices for *true* "active array variables," i.e., the set of array variables defined in assignment statements of the loop body and active if the boolean condition evaluates to be true. Id. The elements of S_F include values generated responsive to indirect loop indices for *false* "active array variables," i.e., the set of array variables defined in assignment statements of the loop body and active if the boolean condition evaluates to be false. Id. Note that SA may equal S_T or S_F, so that the number of arrays for the indirectly indexed access pattern is effectively two. See present application, table 14 ($S_A = S_T$). It follows that if the loop includes no boolean conditions, there is merely one array for the indirectly indexed access pattern.

In one disclosed embodiment, individual array values are computed for respective iterations of the loop based on proxy values, which are respectively unique prime numbers, where the proxy values are substituted for the indirect loop indices. See present application, page 10, lines 23 -29 (describing arrays S_A, S_T and S_F). See also, for example, page 18, line 8, through page 22, line 5 (example 1, showing example of substituting prime numbers for indirect loop indices, among other aspects).

In this specifically described embodiment of the present application, the previously mentioned efficiency relates at least partly to how the indirectly indexed access pattern is structured and generated, which relates to the way the values for the *S* arrays are structured and how they are generated responsive to indirect loop indices.

Accordingly, Applicant herein amends claim 1 to state that the method includes "calculating by the processor indirectly indexed access patterns for respective iterations of the loop based upon the unique proxy values for the indirect loop index variables, each proxy value being a unique prime number and the calculating being such that the calculated indirectly indexed access patterns have respective numbers of pattern values, wherein none of the respective numbers of pattern values exceeds three regardless of how many statements are in the loop." Claims 10, 11, 17 and 18 are similarly amended.

Thus, it should be understood that according to these claim amended limitations for the present invention there are at most three arrays, i.e., "sets of values," for the indirectly indexed access pattern, and that the quantity of these sets of values is not a function of the number of statements in the loop. Huang does not teach or suggest this, neither alone nor combination with

any of the art of record.. No new matter is added, since the original application provides support for the amendments, as described herein above.

The Office action cites the combination of Hwang and Koblitz regarding the use of prime numbers for factorization into a lock for cryptography. Applicant submits that combination of this and the teachings of Huang does not teach or suggest ""calculating by the processor indirectly indexed access patterns for respective iterations of the loop based upon the unique proxy values for the indirect loop index variables, each proxy value being a unique prime number and the calculating being such that the calculated indirectly indexed access patterns have respective numbers of pattern values, wherein none of the respective numbers of pattern values exceeds three regardless of how many statements are in the loop," as claimed. Furthermore, the combination is merely hindsight reconstruction.

Claims 19-21

In addition, claim 19 is added to further distinguish the invention. Claim 19 depends on claim 1. The new claim includes a limitation stating "wherein if the loop includes a boolean condition for selecting active and inactive statements defined in assignment statements of the loop body, such an indirectly indexed access pattern for a respective one of the iterations *consists of* i) a first pattern value generated responsive to indirect loop indices of array variables for such statements active if the boolean condition evaluates to be true and responsive to indirect loop indices of array variables for such statements active if the boolean condition evaluates to be false, ii) a second pattern value generated responsive to indirect loop indices of array variables for such statements active if the boolean condition evaluates to be true, and iii) a third pattern value generated responsive to indirect loop indices of array variables for such statements active if the boolean condition evaluates to be false, and wherein if the loop does not include a boolean condition for selecting active and inactive statements defined in assignment statements of the loop body, the indirectly indexed access pattern *consists of* a single pattern value generated responsive to indirect loop indices of all array variables defined in assignment statements of the loop body" (emphasis added). New claims 20 and 21 with similar language are also herein submitted, depending upon claims 17 and 18. Huang does not teach or suggest this, neither alone nor combination with any of the art of record.. No new matter is added, since the original application provides support for the amendments, as described herein above.

Thus, it should be understood that according to these claim limitations for the present invention not only that there are at most three values for each indirectly indexed access pattern,

and not only that these sets of values are not a function of the number of statements in the loop, but also that there may only be one value for such an indirectly indexed access pattern, depending on a boolean conditional. Contrast this to the increasing number of arrays for increasing numbers of statements, as taught by Huang. Indeed, according to Huang, not only does the number of arrays tend to increase with an increasing number of statements, but the number of arrays may even increase much more than in linear relation to the number of statements.

It should be further appreciated that not only are the number of statements in the loop not a factor influencing the quantity of arrays for the indirectly indexed access pattern according to the above referenced embodiment of the present invention, but also that the small number of arrays (including possibly only one array) apply even if the loop includes a *boolean condition* for selecting active and inactive statements. This also distinguishes the present invention from Huang.

Rejections Under 35 U.S.C. 103 (a)

Claims 3, 4, and 13 are rejected 35 U.S.C. 103(a) as being unpatentable over Huang et al. in view of Tanenbaum, "Structured Computer Organization" ("Tanenbaum"). Claims 3, 4 and 13 are herein canceled.

Claims 5-7 and 12 are rejected 35 U.S.C. 103(a) as being unpatentable over Huang et al. in view of Koblitz, Algebraic Aspects of Cryptography ("Koblitz") and further in view of Hwang, "A New Access Control Method Using Prime Factorization" ("Hwang"). Claims 5 and 12 are herein canceled. Claims 6 and 7 are herein amended to conform them to the amended claim 1. Claims 22 and 23 are herein added, are similar to claims 6 and 7, and depend upon claim 17. Claims 24 and 25 are herein added, are similar to claims 6 and 7, and depend upon claim 18.

Regarding claims 6 and 7, the Office action cites the combination of Hwang and Koblitz for the use of prime numbers for factorization into a lock for cryptography. However, claim 6 concerns pattern values for respective iterations that are calculated by forming products of the proxy values of the indirect loop index variables of the loop for the respective iterations. Applicant submits that the use of prime numbers for factorization into a lock for cryptography does not suggest application to pattern values for respective iterations in analysis and performance of parallel execution of computer instruction loops. The combination of Hwang and Koblitz with Huang is mere hindsight reconstruction.

Claim 7 concerns an indication of a cross-iteration dependency if a greatest common divisor between two indirectly indexed access patterns for two respective iterations is greater than one. The use of prime numbers for factorization into a lock for cryptography does not suggest application to an indication of a cross-iteration dependency for respective iterations in analysis and performance of parallel execution of computer instruction loops. The combination of Hwang and Koblitz with Huang is mere hindsight reconstruction.

PRIOR ART OF RECORD

Applicant has reviewed the prior art of record cited by but not relied upon by Examiner, and submits that the invention is patentably distinct.

REQUESTED ACTIONS

For the reasons explained herein above, Applicant contends that the claims as amended herein are patentably distinct and hereby requests that Examiner grant allowance and prompt passage of the application to issuance.

Respectfully submitted,

Anthony V S England

Anthony V. S. England
Attorney for IBM Corporation
Registration No. 35,129
512-477-7165
a@aengland.com